Nearest-Neighbour Search in High Dimension and Molecular Clustering

Frédéric Cazals
Algorithms project, INRIA Rocquencourt

June 30, 1997

[summary by Frédéric Cazals]

1. Introduction and prerequisites

1.1. **Problem statement.** Given a set of points $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$, the nearest-neighbour (NN) and k nearest neighbours (k-NN) problems can be stated as follows: pre-process P in order to return as fast as possible the nearest or k nearest neighbour(s) of an arbitrary point q according to any Euclidian metric $d(p,q) = (\sum_{i=1}^{d}(p_i - q_i)^2)^{1/2}$. A weakened version of the NN problem consists in returning a point p' which ε-approximates the NN p of q in the sense $d(p',q)d(p,q) \leq 1+\varepsilon$ for any $\varepsilon > 0$. If one denotes $p_{i_1}, \ldots, p_{i_n}$ the points of P sorted by increasing distance to q, an equivalent formulation for the k-NN problem consists in returning a subset $S = \{s_1, \ldots, s_k\}$ with $d(q, s_j) \leq (1+\varepsilon)d(q, p_{i_j})$ for $j = 1, \ldots, k$.

The naive algorithm to compute the NN of a point q consists in checking all the points of P and returning the closest, which has complexity $O(dn)$. On the other hand, the most sophisticated algorithms known until recently had complexities in $O(exp(d)\log n)$ with $exp(d)$ a function growing at least as quickly as 2^d—see e.g., [1]. So that whenever $d \geq \log n$ nothing better than the brute force method was known!

Kleinberg's break-through [4] has been to get around the exponential difficulty by an heavy use of random sampling aiming at "comparing" the points of P through their projections on random lines passing through the origin rather than decomposing the d-dimensional space containing them. The first result is an algorithm returning an approximation of the k-NN in a deterministic way but with an exponential time/space pre-processing. The second algorithm returns an approximation of the NN in a randomised way but with a polynomial pre-processing only. This talk presents these two algorithms and discusses their potential use to a clustering problem arising in chemistry—see section 4.

1.2. Prerequisites.

1.2.1. *A geometric lemma.* The core idea of Kleinberg's method lies in the following property:

Lemma 1. *Let x and y be two vectors of $\mathbb{R}^d$ such that $\| y \| / \| x \| \geq 1+\gamma$ with $\gamma \leq 1/2$. Then, if v is a random vector on the unit sphere S^{d-1} we have* $\Pr[| x \cdot v | \geq | y \cdot v |] \leq 1/2 - \gamma/3$.

Intuitively, short vectors "should not defeat too often" longer ones when comparing their projection on a random line determined by a vector on S^{d-1}. In order to compare two vectors from their projections, the key point is therefore to use a large enough set of lines to capture the probabilistic property contained in the above theorem.

1.2.2. *Empirical measures and Vapnik-Chervonenkis bounds.* Let $(\Omega, \mathcal{F}, \mu)$ be a probability space, $S \subset \mathcal{F}$ a set of events, and $X_1, X_2, \ldots, X_n$ n random variables following the law μ. If one calls the empirical measure of an event s the fraction of X_i's falling into s, the quantity

$$\Delta_S^n = \sup_{s \in S} \left| \frac{1_s(X_1) + \cdots + 1_s(X_n)}{n} - \mu(s) \right|$$

measures the maximum difference over the class S between the empirical measure and the probability. It is a random variable and Vapnik-Chervonenkis's contribution [5] has been to elucidate the conditions under which it converges in probability to zero, that is the conditions under which $\lim_{n \to \infty} \Pr[\Delta_S^n > \varepsilon] = 0$ for any ε. To sketch this contribution, let a range-space be a couple $(\mathcal{P}, \mathcal{R}) = \big((\Omega, \mathcal{F}, \mu), \mathcal{R} \subset \mathcal{F}\big)$. We shall say that a set A of finite cardilality is shattered by $\mathcal{R}$ if $\forall a \in 2^A \quad \exists r \in \mathcal{R}$ such that $a = r \cap A$. The dimension of Vapnik-Chervonenkis of $(\mathcal{P}, \mathcal{R})$ is the cardinality of the biggest $A \subset \Omega$ shattered by $\mathcal{R}$.

Definition 1. A γ-sample for $(\mathcal{P}, \mathcal{R})$ is a finite set $A \subset \Omega$ such that $|\mu(r) - \mid r \cup A \mid / \mid A \mid| \leq \gamma$, $\forall r \in \mathcal{R}$.

Theorem 1 ([5]). *For a range space of dimension k, a random sample of size $l \geq \frac{16}{\gamma^2}(k \log \frac{16k}{\gamma^2} + \log \frac{4}{\delta})$ is a γ-sample with a probability at least $1 - \delta$.*

1.2.3. *Exceptional and ρ-distinguishing sets.* As pointed out above, we are interested in comparing points with respect to their projections on vectors from S^{d-1}. For two vectors x and y with $\| y \| / \| x \| \geq 1 + \gamma$ we call their exceptional set

$$W_{x,y} = \{v \in S^{d-1} \text{ such that } \mid x \cdot v \mid \geq \mid y \cdot v \mid\}.$$

And a random set of vectors V from S^{d-1} is called ρ-distinguishing if

$$\forall W_{x,y}, \quad \mu(W_{x,y}) < \rho \Longrightarrow \mid V \cap W_{x,y} \mid / \mid V \mid < 1/2.$$

More prosaically, a set V is ρ-distinguishing if a majority of its points do not fall into some exceptional set of size smaller than ρ.

1.2.4. *Hyper-planes arrangements.* An arrangement of n hyper-planes in $\mathbb{R}^d$ is said to be in general position if any d hyper-planes have a unique point in common, and any $d+1$ hyper-planes do not share a point. Given a hyper-plane h and a point p, p is either above, on, or below h, which is called its position. A face on an arrangement is the set of points having the same position with respect to all the hyper-planes. The dimension of a face is its affine dimension. It is known from [2] that

Theorem 2. *The number of d-faces of an arrangement of n hyper-planes in general position is $f_d(n) = \sum_{i=0}^{d} \binom{n}{i}$.*

1.2.5. *Digraphs.* A complete digraph G on n vertices $1, 2, \ldots, n$ is a directed graph which contains for any pair of vertices $\{i, j\}$ either the edge (i, j) or (i, j). An apex of G is a vertex with a directed path of length at most two to any vertex. At least, an apex ordering of G is an ordering $i_1, \ldots, i_n$ of its vertices such that i_k is an apex for the sub-digraph $G[i_k, i_{k+1}, \ldots, i_n]$. The following is straightforward:

Theorem 3. *Every n-node complete digraph has an apex ordering computable in $O(n^2)$.*

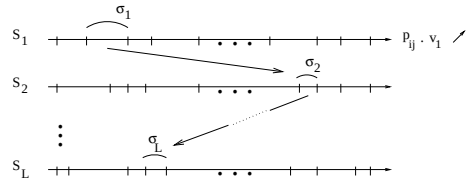

FIGURE 1. Traces

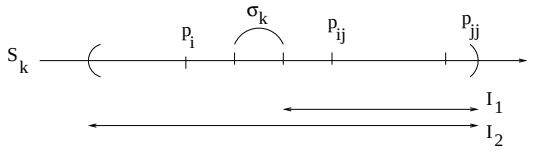

FIGURE 2. σ-domination

1.3. **First results.** The following theorems can be proved:

Lemma 2. *Let μ be the uniform measure on S^{d-1}. The dimension of the range-space*

$$((S^{d-1}, \mathcal{F}, \mu), \{W_{x,y} \mid \mu(W_{x,y}) \leq \rho\})$$

is less than $d' = 8(d+1)\log(4d+4)$.

Lemma 3. *There exists c_0 such that a random sample of S^{d-1} of size $f(\delta,\gamma)$ is a $\gamma/2$-sample for the range-space $((S^{d-1}, \mathcal{F}, \mu), \{W_{x,y} \mid \mu(W_{x,y}) \leq \rho\})$ with $f(\delta,\gamma) = \frac{c_0}{\gamma^2}\left(d' \log \frac{d'}{\gamma^2} + \log \frac{1}{\delta}\right) = \theta(d \log^2 d)$.*

Corollary 1. *A set V of $f(\gamma,\delta)$ vectors from S^{d-1} is $(1/2-\gamma)$-distinguishing with a probability at least $1-\delta$.*

2. First algorithm

2.1. **Construction of the data structure.** This algorithm returns an approximation of the k-NN of a point q. To build the data structure from which it does so, we first draw uniformly at random a set V of $L = f(\frac{\varepsilon}{3}, \delta) = \theta(d \log^2 d)$ vectors from S^{d-1}. Then for each vector $v_l \in V$, the following is done: 1. compute $v_l \cdot p_{ij}$ with $p_{ij} = (p_i + p_j)/2$, $1 \leq i, j \leq n$; 2. sort the p_{ij} according to the values of $v_l \cdot p_{ij}$ and denote S_l the list obtained. The list of lists $S_1, \ldots, S_L$ is denoted Σ.

For each such list, a pair of consecutive entries is called a primitive interval, and a sequence of primitive intervals is called a trace—see Figure 1. The maximum number of traces is upper-bounded by $(n^2)^L = n^{O(d \log^2 d)}$. But a trace is realizable if

$$\exists q \in \mathbb{R}^d, \forall k = 1, \ldots, L, v_k \cdot p^{(k)}_{i_1 i_2} < v_k \cdot q < v_k \cdot p^{(k)}_{i_3 i_4}.$$

So that realizable traces are defined with respect to the Ln^2 hyper-planes $v_k \cdot (p^{(k)}_{i_1 i_2} - x)$. And from theorem 2, the number of such traces is $\sum_{i=0}^{d} \binom{Ln^2}{i} = O(n \log d)^{2d}$.

Definition 2. For a realizable trace $\sigma = \sigma_1 \cdots \sigma_k \cdots \sigma_L$, p_i is said to σ-dominate p_j in S_k if p_{ij} lies in the interval $(\sigma_k, \ldots, p_{jj})$.

For each realizable trace σ, the construction is as follows: (1.) build a complete digraph G_σ on $\{1, 2, \ldots, n\}$ with the edge (i,j) if p_i σ-dominates p_j in half of the lists of Σ; (2.) build an apex ordering (σ, G_σ) of the nodes of G_σ.

The idea behind the domination definition is depicted on Figure 2: if p_{ij} falls in the desired interval denoted I_1, then $p_i \in I_2$ and we have $\mid v_k \cdot (p_i - q) \mid < \mid v_k \cdot (p_j - q) \mid$.

2.2. **Algorithm.** To process a query associated to a point q: 1. compute $\sigma(q) = \sigma_1(q) \cdots \sigma_L(q)$ with $\sigma_k(q)$ the primitive interval from S_l containing $v_l \cdot q$; 2. retrieve the apex ordering associated to $\sigma(q)$ and return the k first entries.

This algorithm therefore returns an ε-approximation of the k nearest neighbours of a point q in a deterministic fashion, so that the answer is guaranteed to be correct if the random sample V is actually $(1/2 - \varepsilon/3)$-distinguishing—see §1.2.3 and Corollary 1. Roughly speaking, the correctness

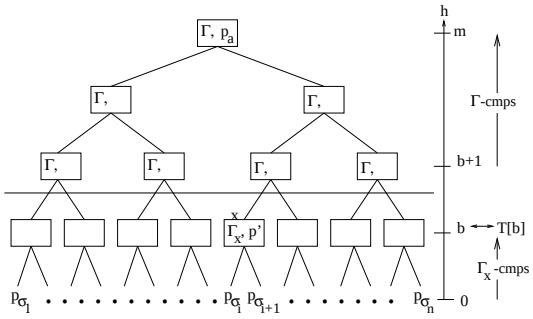

FIGURE 3. Tournament tree

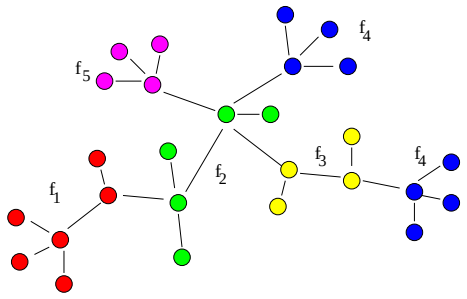

FIGURE 4. Molecule and fragments

of the algorithm comes from the fact that if a non-desired point p_1 has been returned instead of a desired point p_2, then p_1 dominated p_2 in more than half of the lists of Σ, and the sample V was not distinguishing enough with respect to the exceptional set $W_{q-p_1,q-p_2}$.

The pre-processing requires $O\left(Ln^2(n\log d)^{(2d)}\right)$ time and $O\left(n(n\log d)^{(2d)}\right)$ space. The cost of a query is $O(k+L(d+\log n))=O(k+(d\log^2 d)(d+\log n))$.

3. Second algorithm

As opposed to the first algorithm, the second one does not try to compute a partition of the requests' space and proceeds in two steps. The first one consists in drawing a random sample from P of the appropriate size and returning the closest point to q. The second one compares iteratively q to pairs of points of P in a tournament way depicted on Figure 3. Assuming that $n=2^m$, the overview of this second stage is the following:

1. a random sample V of $\Theta(d\log^2 n(\log^2 d+\log d\log\log n))$ vectors is drawn uniformly on S^{d-1}, and a multi-set Γ_v of V is assigned to each internal node v of the binary tree whose leaves are a permutation of the points of P. For a query point q, two points p_i and p_j of P, and a multi-set Γ_v, p_i is said to dominate p_j if $\mid v_k\cdot(p_i-q)\mid<\mid v_k\cdot(p_j-q)\mid$ holds for a majority of vectors v_k in Γ_v. Otherwise p_j dominates p_i;
2. each internal node v of the tree is assigned its dominating child for the multi-set Γ_v.

The point eventually returned is the best candidate from the two points returned by the two steps. It can be shown that an ε-approximation is returned with a probability greater than $1-\delta$ in $O(n+d\log^3 n)$ time with a space requirement of $n\mid V\mid$.

4. Application to molecular clustering

Suppose we are given a set of d molecular fragments, say $F=\{f_1,\ldots,f_d\}$ and a set $M=\{m_1,\ldots,m_{N_m}\}$ of N_m molecules, each described by a set of fragments of F—see Figure 4. We shall represent a molecule m_i by a sequence of d boolean values $m_i=b_{i,1}b_{i,2}\cdots b_{i,d}$ with $b_{i,j}=1$ if m_i contains f_j and $b_{i,j}=0$ otherwise. This formulation fails to report multiple occurrences of a given fragment in a molecule, but it has the nice property that a molecule is represented by a point on the hyper-cube $H^d=\{0,1\}^d$. Given two molecules, we call their *similarity* the number of common fragments that is the quantity $\mathrm{sim}(m_i,m_j)=\sum_{k=1}^{d}b_{i,k}\cdot b_{j,k}$.

Given the set M we are interested in the following problem: find a partition of M into subsets of neighbours or clusters. To do so, one way to proceed see [3] consists in first building a Minimum Spanning Tree on the input data set, second removing those "too long" edges from the MST, and third computing the connected components we are left with. The key point lies in the MST computation, and it is shown in [6] that

Theorem 4. *A MST on n points in dimension d can be found in $O(2^d\ n^{2-1/2^{d+1}}(\log n)^{1-1/2^{d+1}})$ time.*

Unfortunately for our concern where d can range from 500 to 2000, the 2^d constant is prohibitive. Kleinberg's algorithms customised to the hyper-cube setting could make Yao's algorithm interesting in practice.

References

[1] Arya (S.), Mount (D. M.), Netanyahu (N. S.), Silverman (R.), and Wu (A. Y.). – An optimal algorithm for approximate nearest neighbour searching. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. pp. 573–582. – New York, 1994.

[2] Edelsbrunner (Herbert). – *Algorithms in combinatorial geometry*. – Springer-Verlag, Berlin, 1987, *EATCS Monographs on Theoretical Computer Science*, vol. 10, xvi+423p.

[3] Jain (Anil K.) and Dubes (Richard C.). – *Algorithms for clustering data*. – Prentice-Hall Inc., Englewood Cliffs, NJ, 1988, *Prentice-Hall Advanced Reference Series*, xiv+320p.

[4] Kleinberg (J.). – Two algorithms for nearest-neighbour search in high dimension. In *ACM STOC*. – El Paso, Texas, USA, 1997.

[5] Vapnik (N.) and Chervonenkis (A.). – On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, vol. 16, n° 2, 1971, pp. 264–280.

[6] Yao (Andrew Chi Chih). – On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, vol. 11, n° 4, 1982, pp. 721–736.